SOFTWARE MAINTENANCE: RESEARCH AND PRACTICE, VOL. 9, 203–216 (1997)

Practice

Maintenance of COTS-intensive Software Systems

DUANE W. HYBERTSON,* ANH D. TA AND WILLIAM M. THOMAS
The MITRE Corporation, 1820 Dolley Madison Blvd., McLean, VA 22102, U.S.A.

SUMMARY

The software industry has made extensive use of commercial software tools such as compilers and editors in development environments of computer-based systems for several decades. However, in recent years an emerging trend is the extensive use of commercial off-the-shelf (COTS) software products as a major part of delivered software systems. It is generally recognized that this trend introduces a significant change to the development of computer-based systems. The thrust of this paper is that this trend also introduces a significant change to the software maintenance process. The paper addresses some of the issues involved in the maintenance of COTS-intensive software systems, and the reasons why the COTS factor constitutes a significant change from traditional software maintenance processes. Finally, some lessons learned are offered as suggestions for addressing the issues in the maintenance of COTS-intensive systems. © **1997 by John Wiley & Sons, Ltd.**

J. Softw. Maint., **9**, 203–216 (1997)

No. of Figures: 6. No. of Tables: 1. No. of References: 10.

KEY WORDS: commercial off-the-shelf software; maintenance processes; maintenance management; maintenance organization; software evolution; technology change

1. INTRODUCTION

The use of commercial software and associated technologies has gradually increased over the 40-year history of application software systems. Types of commercial software have expanded from the initial base of operating systems, compilers and editors to database management systems (DBMSs), network and communications software, word processors, spreadsheets, graphical user interfaces (GUIs), application-specific generators and packages, and Web browsers. The role of commercial software has expanded, from development/maintenance tools and elements of the operational environment of application systems to being a direct component of application systems. This increase in the use of commercial off-the-shelf (COTS) products has a significant impact on the development and maintenance of application systems. The term 'COTS-intensive system' in this paper means an application software system composed primarily of multiple COTS products.

* Correspondence to: Duane W. Hybertson, The MITRE Corporation, Mail Stop W624, 1820 Dolley Madison Blvd., McLean, VA 22102, U.S.A. E-mail: dhyberts@mitre.org

CCC 1040–550X/97/040203–14$17.50

Received 14 February 1997
Revised 15 May 1997

To help define the scope of the paper, we will describe our view of COTS software and relate it to the larger context of software reuse. Any software that is used in a system but is not custom coded for that system may be considered reused. Reused software may be obtained from a variety of sources, including one's own organization, a public code library, the government or a commercial vendor. Important distinctions among these sources of reused software include (a) who maintains the reused software, and (b) what forces tend to cause change in the reused software. From this perspective, commercial software (COTS) represents the most extreme difference from the traditional custom software development model. This is because the vendor maintains all the code in a commercial product, and changes in a COTS product are driven by the entire community of users, the vendor and market dynamics (i.e., the market competition).

The differences introduced by extensive use of COTS products in software systems raise new issues in software maintenance. The purpose of this paper is to explore the effect of COTS software and market dynamics on the maintenance of COTS-intensive systems. The aim is to identify issues and suggest short-term actions to cope with these issues until they are resolved by further research and experience. The paper is written from the perspective of an organization that is responsible for the maintenance and evolution of a COTS-intensive system. Note that a vendor of a suite of interacting COTS products could fit this description.

The context of this topic is illustrated in Figure 1. The maintenance of COTS-intensive systems in the lower right part of the box may be viewed as inheriting features from two dimensions: traditional maintenance and COTS-intensive development. Comparisons made with traditional maintenance illustrate how COTS and technology change the maintenance process, while comparisons with COTS-intensive development illustrate similarities and differences between maintenance and development of COTS-intensive systems.

It is understood that both dimensions in Figure 1 (development-maintenance and custom–COTS) represent a continuum—i.e., there is no clearly defined boundary between what may be regarded as maintenance versus development or between a custom versus COTS-intensive system. The continuum between traditional custom and COTS systems is associated with the increasing amount of COTS products in a system, ranging from none

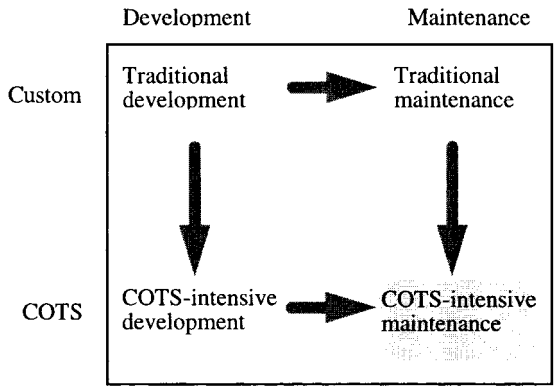

Figure 1. Context of the analysis of COTS-intensive maintenance

to all. The continuum between development and maintenance is associated with the size of changes after deployment, where major changes become much like development.

Nevertheless, in this paper, both dimensions will be treated as dichotomies, because we believe that important and useful distinctions can be made between the process types resulting from the two interacting dimensions (see Figure 1), viz., traditional development, traditional maintenance, COTS-intensive development and COTS-intensive maintenance. In particular, the COTS-intensive maintenance process has properties that should be investigated and discussed. An important premise of the paper is that an application system containing multiple COTS products requires a different kind of maintenance planning from that for a mostly custom system. Figure 2 summarizes our view of traditional software maintenance, drawing upon the IEEE definitions of maintenance types: corrective, adaptive, and perfective (IEEE, 1993). Furthermore, Figure 2 presents the view that technology is only an indirect source of change in traditional software maintenance.

The paper presents a context and foundation for discussing COTS software, then discusses COTS-intensive maintenance issues and potential ways to address these issues. At the current state of maturity of software engineering, the issues and problems of COTS-intensive systems maintenance—even though they have not necessarily been articulated in the literature—are better understood than the solutions.

The organization of the paper is as follows: Section 2 presents issues that are characteristic of the maintenance of COTS-intensive software systems. Section 3 suggests ways to mitigate or solve the issues presented in Section 2 and outlines some ideas for further research. Finally, Section 4 provides conclusions.

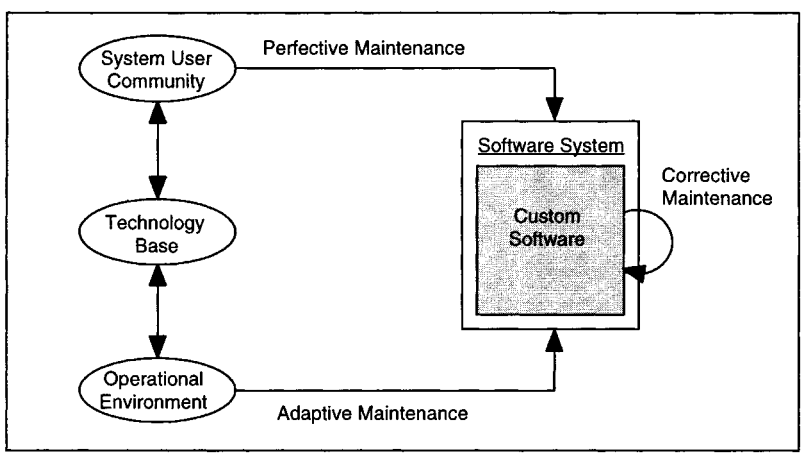

Figure 2. Sources of traditional software change

2. COTS-INTENSIVE SOFTWARE MAINTENANCE ISSUES

2.1. Overview

The crossover from COTS systems development to maintenance may be viewed as a shift from selecting an initial set of available COTS products to a focus on the monitoring and adaptation of evolving COTS products and technology. A COTS-intensive system is directly plugged into technology change, which has both positive and negative consequences for maintenance. This COTS crossover is discussed in Section 2.2. The crossover from traditional maintenance to COTS-intensive system maintenance may be characterized as a shift from a predominantly custom requirements-driven approach that focuses on custom solutions to an approach driven by both requirements and evolving technology. This maintenance crossover is discussed in Section 2.3.

2.2. Applying COTS development context to COTS maintenance

Since maintenance of a system nominally lasts far longer than development, the evolution of COTS products, and even of the technologies on which they are based, become dominant factors that change the focus of maintenance of COTS-intensive systems. Both COTS products and technologies have usage life cycles. This phenomenon, in the general form of diffusion of innovation, has been studied by a number of researchers. A classic work is Rogers' analysis of diffusion of innovation (Rogers, 1983). More recent discussions of this phenomenon include Girifalco (1991) in the context of technology change, and Millett and Honton (1991) from a forecasting perspective. An examination of the time required for diffusion or maturation of software technology was performed by Redwine in the 1980s (Redwine and Riddle, 1985). The discussion that follows is based on these analyses.

The life of an innovation (which in the present context could be a COTS software product or a software technology) begins with adoption and ends with discontinuance. During its period of use, a given product undergoes changes, both major and minor, that are made available in periodic releases. Products in a technology area, such as user interface models or database systems, compete with each other in the market. One product may dominate for a time, then another may become dominant. Within a technology area, technologies have life cycles where one becomes dominant for a time, then is replaced over a period by another technology that becomes dominant (e.g., character-based user interfaces being replaced by GUIs). Standards emerge as part of a maturing technology. A technology enables new categories of products to emerge, or in some cases products lead the emergence of a new technology. In this way there is a relationship between a technology and the products that represent it, and between their respective life cycles—although a technology in general has a longer life than a product.

The general pattern can be modelled as shown in Figure 3, which represents the rising and then falling demand for a technology or a COTS product as a curve on some time scale. The three curves in the figure may represent three different technologies within a technology area, or three different competing COTS products.

Despite the general patterns exhibited by product and technology life cycles, it is very

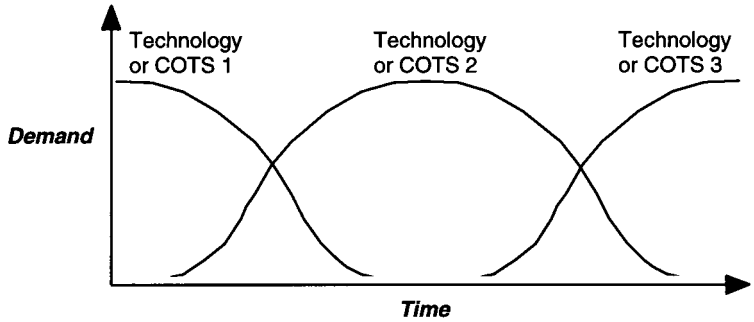

Figure 3. Demand curves for technology or COTS

difficult to predict specific products that will dominate and specific time scales of their evolution. Volatility is a property of COTS products and technology, especially in a relatively immature discipline like software engineering. Volatility results in unpredictability of direction, extent and timing of change, which in turn adds complexity and risk to maintenance planning and decision-making. The direction of change is reflected in the question: 'Which technology (or product) will dominate?' This is a major question during development of a COTS-intensive system, but its effects become visible during maintenance. The extent and timing of change is reflected in the question: 'How much and when will a product change?' This is often a major issue during maintenance.

Within one technology area, there are additional factors in determining the life cycle of one COTS product:

- Standards cycle: standards have a life cycle, which is a factor in each COTS product and therefore in the maintenance of a COTS-intensive system.
- Enterprise planning cycle (budget and technical): an enterprise (such as the customer or system maintainer organization) typically has a planning cycle that may affect its decisions about upgrading and replacing COTS software and technology.

When multiple COTS products are collected in one system, the issues introduced by a single COTS product become magnified. As illustrated in Figure 4, the evaluation of different technologies at a specific time shows that they are often at different demand (and maturity) levels. For example, assuming the scenario drawn in Figure 4, in 1990 character-based user interfaces were in the process of being replaced in the market with GUIs, while at the same time relational DBMSs were dominant and object-oriented DBMSs were just emerging.

Even within one COTS product, changes can cause disruptions. A new version of a COTS product may not be compatible with the old one. A sample scenario of both issues, i.e., within-product changes and multiple-product alignment of changes, is presented in Figure 5 for four COTS products, addressing only product patches and upgrades (not considering product or technology replacement issues).

The problem is reflected in the frequent decisions needed about incorporating patches and upgrades into releases of the COTS-intensive system. Considerable planning and co-

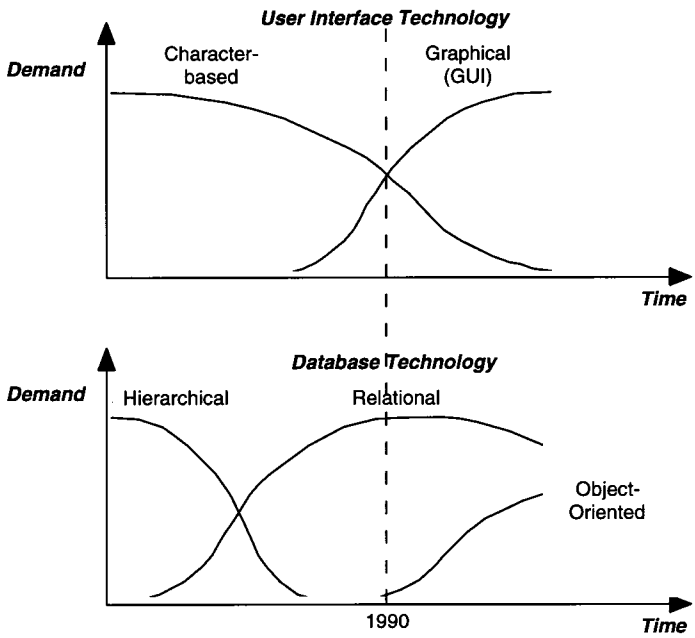

Figure 4. Independent technology area cycles

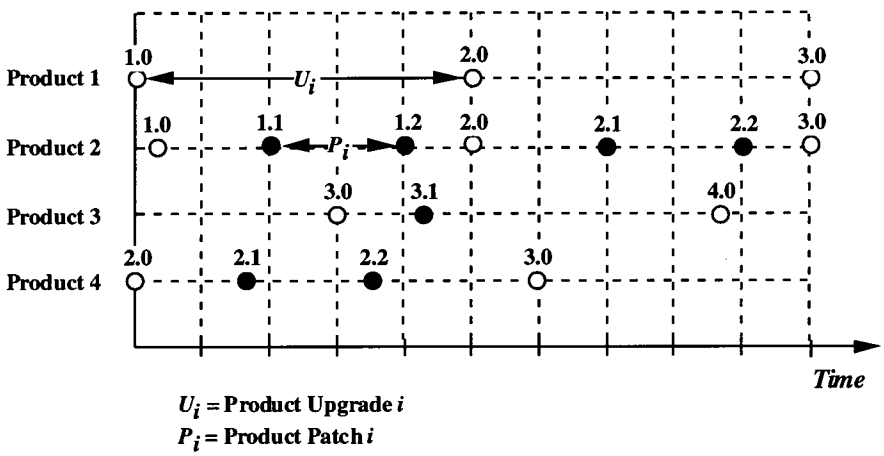

Figure 5. Hypothetical sample of multiple COTS product cycles

ordination is required. Some organizations are moving toward trying to package multiple COTS upgrades together in 'co-ordinated releases' of the system (Slater, 1997).

2.3. Changes from traditional maintenance

This section briefly discusses how COTS-intensive systems affect these traditional maintenance factors: maintenance types, change forces, levels of evolution, laws of evolution, and perceived maintenance problems.

Maintenance types. From the standpoint of maintenance types, the biggest impact of COTS is that COTS products are a major new source of adaptive maintenance—not just adapting to the environment, but adapting the system to changes of other COTS software that is part of the system itself. No new maintenance type appears to be needed for COTS-intensive systems.

Change forces. The most significant difference in the forces for change introduced by COTS (compared with a traditional system as shown in Figure 2) is the direct influence that technology and COTS products have on the application system. In traditional development, the technology base of a system played a smaller, supportive role, and was largely isolated from the application software system evolution. In COTS-intensive systems, the technology base is a constituent element of the system, and its evolution becomes a direct influence for change in the system, as shown in Figure 6.

Levels of evolution. In practice the selection of multiple COTS products is influenced by consideration of both viability and interactions among products. Viability includes questions of which technologies, vendors and COTS products will be significant and continuing players in the market. Interaction issues include integrability of potential COTS selections, possible exploiting of the practice of bundling multiple COTS into a single suite in which the vendor presumably does the integration, and consistency of standards or standards profiles adhered to by the various COTS products. Criteria such as adhering to open system interfaces and industry standards, play a role in these decisions. From the standpoint of maintaining a given system with its current and future requirements, we see three levels of possible change in a COTS-intensive application system:

- *Evolution/upgrade of a product*. There is a life cycle associated with every product

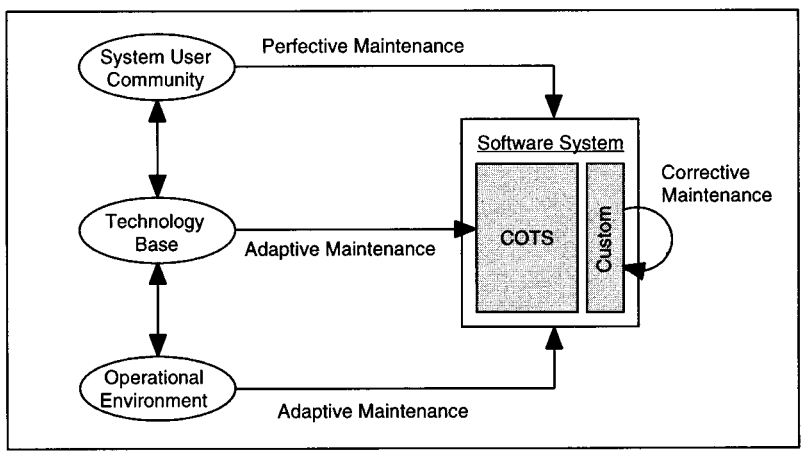

Figure 6. Sources of COTS-intensive system change

sold by a vendor. Upgrades can be minor (patches) or major new capabilities. Upgrades occur relatively frequently. An upgrade may be incorporated due to any of these: it fixes an error; it adds a feature the application system needs; or the vendor will stop supporting the old version. Associated with this is the notion of a product family, which collectively has a longer life cycle, but each member of a product family typically has its own life cycle.

- *Replacement of product A with product B in the same technology*. This infrequently occurs in maintenance, but may become more frequent with increasing standardization and open systems. A product may be replaced for any of these reasons: the new product conforms to a desired standard or provides a major function that the application system needs; the new product is becoming dominant in the market; or support for the existing product is becoming unacceptable (e.g., if the vendor stops support for the product.)
- *Replacement of technology A with technology B in the same technology area* (e.g., replace relational DBMS with object-oriented DBMS). This is significant re-engineering. It may be done in maintenance if the new technology would make the application system considerably better or solve significant problems, or is becoming dominant (although the latter is usually not sufficient by itself for this large step).

The relation of these levels to software release practices may be delineated as follows and as summarized in Table 1:

- A COTS product upgrade in the form of a *minor* release is analogous to the traditional program release level, but in the form of a *major* release is analogous to the traditional program release sequence level.
- Replacing a COTS product does not have a direct analogue in traditional maintenance. However, if the old and new COTS products are quite similar, the situation can be analogous to the traditional program release sequence level.
- Replacing COTS technology is analogous to replacing one generation of technology with a new generation in traditional maintenance.

Laws of evolution. Several-well known laws of evolution have been derived from the empirical investigation of large software programs (Lehman and Belady, 1985; Sommer-

Table 1. Levels of maintenance comparison

Traditional	COTS-intensive		
	Upgrade	Replace product	Replace technology
Release	Minor release		
Release sequence	Major release	Similar product	
Generations			Different technology

ville, 1992). Additional related studies are reported by Kemerer (1995). These laws are summarized here from Lehman and Belady (1985, Chapter 19):

1. Continuing change: A program that is used undergoes continual change or becomes progessively less useful.
2. Increasing complexity: As an evolving program is continually changed, its complexity increases, reflecting a deteriorating program structure.
3. Invariant attributes: Program evolution dynamics make system attributes such as size, time between releases, and number of reported errors approximately invariant for each system release—an attribute sometimes known as the fundamental law of program evolution.
4. Invariant work rate: As an aspect of the conservation of organizational stability during the active life of a program, the program's rate of change is approximately constant and independent of the resources devoted to maintenance.
5. Limit of change in each release: During the active life of a program, the program develops a characteristic average increment of safe change per release (thus conserving familiarity) that if exceeded causes quality deterioration and usage difficulties.

We can therefore well ask what is the effect of COTS on the laws of evolution? Two perspectives are useful for answering this question. First, we offer some observations from the perspective of each *individual COTS product*:

- Laws 1–4 (continuing change, increasing complexity, invariant attributes, invariant work rates) are about issues that concern the COTS vendor much more directly than the system maintenance team or the system users. However, a COTS product reflects the technology era in which it was developed, and when that technology context changes significantly, it is difficult for the vendor to change the product to keep up. New replacement products become a more viable option at this point.
- Law 5 (limit of change in each release) affects both the system maintenance team and the system users. The system maintenance team must reintegrate an updated COTS product. The reintegration is affected by how much the product changes in a release. The system users must adapt to the visible changes in a new release of the system, including changes in individual COTS products that are part of the system.

Second, from the perspective of the *overall system*, the general observation is that more research and experience are needed to determine the applicability of the laws of evolution to COTS-intensive systems. However, the following observations may be made concerning the system as a whole:

- The deterioration reflected in Laws 1 and 2 is a cause for concern, because of the increased pressure for change due to inclusion of COTS products. If these laws hold for COTS-intensive systems, then these systems may potentially deteriorate more rapidly than traditional systems due to more disjointed (and possibly increased amount of) change coming from multiple independent vendors.

- The increased rate of change of each product and the cumulative cycles of the three levels of evolution, multiplied by many products (at least one for each technology area in the system) has the effect of adding significantly to the complexity, volatility and unpredictability of the maintenance planning situation. This factor affects both the system maintenance team and the system users, especially if Law 5 applies.

Perceived maintenance problems. Lientz and Swanson (1981) identified the top three perceived maintenance problems as:

1. demand for enhancements and extensions,
2. competing demands for programmer time, and
3. documentation quality.

These problems appear to be ones that COTS-intensive systems may help alleviate. The traditional demand by users for enhancements will likely be satisfied to some degree in a COTS-intensive system by the upgrading of COTS products as the technology evolves. Issues regarding demands for programmer time and documentation quality now shift more to the COTS vendors and away from the system maintenance organization, although the maintenance organization now faces other staffing issues as described in the next section.

However, while a COTS-intensive solution may alleviate some of the workload in the maintenance organization (for example, many of the enhancements are produced by the COTS vendor), it is likely to add some new problems. The issue of keeping up with the *demand* for enhancements and extensions may be replaced by difficulty in keeping up with the *supply*. In a system consisting of numerous COTS products, there is likely to be a continuing supply of upgrades to be integrated, adding to the difficulty in keeping the system up to date. From a user perspective, the cycle-time issues may still be a concern, particularly because in a COTS-intensive system control over much of the maintenance activity (e.g., prioritization of change requests) is exercised by the vendors. Thus, the maintenance organization may not be as timely in meeting user demand for change.

2.4. Discussion of major changes in COTS-intensive maintenance

Differences in the underlying maintenance situation with COTS-intensive systems, as opposed to traditional maintenance, can be grouped into two broad categories. The first category is associated with the system maintenance organization, and the second is associated with issues driven by the larger context of COTS maintenance.

2.4.1. System maintenance organization

1. *System maintenance team skills*: skills needed to monitor and integrate COTS product upgrades or replacements into an application system may differ from those of the traditional maintenance function of analysing and modifying custom source code. This is not necessarily positive or negative, but is important to recognize for planning, staffing and training purposes.

2. *Infrastructure/organization*: maintenance of an in-house support group is typically needed to administer each product or product family. This is an additional cost, but may be shared with other projects or systems across the larger enterprise containing the system maintainers for jointly used big-ticket products, such as DBMSs.
3. *COTS maintenance cost*: COTS costs include purchase cost, ongoing licence and upgrade fees and product training. From an overall life cycle system acquisition perspective, considerable cost is shifted from developing custom code to licence and product maintenance fees, which may add to the maintenance cost. This issue affects users as well as the system maintenance organization.

2.4.2. Larger context

1. *Larger user community*: COTS users are part of a larger community of users which can be a resource (including user groups). This is a positive factor when the needs of the larger community are consistent with the needs of a specific system, but when these needs are inconsistent or contradictory, the effect is likely to be reduced support. Being part of a community means less control over changes and improvements to COTS products than there is over custom products.
2. *Modernization*: COTS vendors help the system keep pace with technology growth and help modernize on a more-or-less continual basis. The positive side of this is that it helps keep the system from becoming obsolete. The negative side is the cost and risk of making changes, including some significant ones, even when the application may not require the changes. Product upgrades are driven by technology, need (user community) and vendor economics.
3. *Split maintenance function*: each COTS product is maintained by its vendor (which is in effect an 'outsourcing' of the maintenance function and splitting into multiple pieces), while the overall system architecture and integration is maintained by what was traditionally the single software maintenance team. Thus, there are multiple, independent maintenance teams. The positive side of this is the additional maintenance help the system maintainer receives from the vendors. The negative side is that the different pieces are not co-ordinated; versions of each product may differ from the direction of the overall system, and the products (vendors) may go off in all directions with respect to functionality or standards.
4. *More complex planning*: when multiple technologies and COTS products are combined in one system, the unpredictability and risk of change is multiplied, and planning becomes very complicated. Synchronization among a large collection of vendors may be difficult.

3. POTENTIAL SOLUTIONS

Many of the issues raised above will require more research and experience to develop accepted solutions. However, some actions can be taken. Preliminary suggestions include:

1. *Consider a new personnel capability mix*: maintenance of COTS-intensive systems evidently calls for a different kind of maintenance team: a shift from programmers analysing and changing source code to analysts maintaining architecture and tracking technology/COTS. In addition to support functions such as configuration management, three types of people on the system maintenance team are suggested: (1) system architects, (2) technology specialists covering each technology area, COTS product and related standards represented in the application system, and (3) integration and test specialists. Some people may serve in two or all three of these roles, which would help the communication and integrity of the operation.
2. *Increase prototyping of potential product changes*: set up a laboratory with an operational testbed instantiated with the application system to prototype potential changes (which may involve one or many product changes). Suggested procedure: if possible, separate each change (e.g., one COTS product upgrade) and test it; if it passes, then add one more change, etc., until the complete set has been tested and certified in the testbed, then release the new application system version with the (multiple) changes. But, the situation is more complicated if the application system is used by multiple organizations or at multiple sites: the system maintenance organization may have to determine which changes can be optional to allow certain enterprises to not upgrade certain products (some of the changes in the new system) because of their policies or standards.
3. *Implement new tracking and assessment activities*: modest tracking of each technology and vendor/product may be a good approach. Potential tracking heuristics include:
   - *Individual COTS patches*: patches are less important to install, but also less risky and disruptive, and they generally correct errors; if the users need the errors fixed, install it, otherwise defer to other factors.
   - *Individual COTS upgrades*: upgrades are usually more important to install sooner or later, if the users intend to stay with the product for an extended period.
   - *COTS replacement*: pursue only if the product is not performing adequately or is lagging behind what other competitors are doing, or the vendor is in serious danger of disappearing.
   - *Individual technology*: track via the COTS in the application system, but also competing products in that technology (e.g., WordPerfect versus Word, or Netscape versus Internet Explorer).
   - *Multiple COTS*: while tracking, consider whether the upgrade will have a synergistic or contradictory effect with other products in the system. This drives the necessity for the additional prototyping activities discussed in item 2.
4. *Actively participate in standards activities*: support and actively participate in standards that are related to the COTS products in the system, to encourage other users and vendors to move toward standardization and open systems. This will give more options for upgrading and modernization because more products should be interoperable and interchangeable.

With further development of ideas such as these, one can avoid the classic problem of 'chasing technology' while at the same time avoid depending on obsolete technology.

More generally, developing COTS maintenance solutions will help achieve the simultaneous goals of reducing change risks, maintaining system usefulness and making maximum use of maintenance resources. Further research is needed to determine an appropriate decision approach for maintaining COTS systems with varying budget levels, including especially the case of a minimal budget. Research is also needed to answer the question of whether COTS software and COTS-intensive systems follow the traditional laws of program evolution. In general, more experience, as well as research with COTS-intensive systems, is needed to develop solutions and expand them into a systematic approach for maintaining this type of system.

4. CONCLUSIONS

We summarize the lessons we have learned about the maintenance of COTS-intensive systems as follows:

1. COTS use is increasing, and maintenance issues of COTS-intensive systems need to be articulated and addressed.
2. Extensive use of COTS products introduces significant new factors more directly into the maintenance situation, including vendors, market competition and increased pressure for change. These factors are likely to cause changes in the planning and management of software maintenance, including skills needed, type of activities and relationships among the stakeholders of software that is in operation.
3. Until more definitive information is available, some suggested actions can be taken:
   - consider a new personnel capability mix,
   - increase prototyping of potential product changes,
   - implement new tracking and assessment activities, and
   - actively participate in standards activities.
4. There is some understanding of what the changes and issues are, but more research and more experience are needed to establish the precise nature of the COTS-intensive maintenance situation and how to manage it.

References

Girifalco, L. A. (1991) *Dynamics of Technological Change*, Van Nostrand Reinhold, New York, NY, 524 pp.

IEEE (1993) *IEEE Standard for Software Maintenance*, IEEE Std 1219B1993, Institute of Electrical and Electronic Engineers, Inc., New York, NY, 39 pp.

Kemerer, C. F. (1995) 'Software complexity and software maintenance: a survey of empirical research', *Annals of Software Engineering*, **1**(1), 1–22.

Lehman, M. and Belady, L. (1985) *Program Evolution: Processes of Software Change*, Academic Press, London, 538 pp.

Lientz, B. P. and Swanson, E. B. (1981) 'Problems in application software maintenance', *Communications of the ACM*, **24**(11), 763–769.

Millett, S. M. and Honton, E. J. (1991) *A Manager's Guide to Technology Forecasting and Strategy Analysis Methods*, Battelle Press, Columbus, OH, 99 pp.

Redwine, S. T. and Riddle, W. E. (1985) 'Software technology maturation', in *Proceedings of the*

8th International Conference on Software Engineering, IEEE Computer Society Press, Los Alamitos, CA, pp. 189–200.
Rogers, E. M. (1983) *Diffusion of Innovations*, 3rd edition, The Free Press, New York, NY, 453 pp.
Slater, D. (1997) 'Smooth moves: client-server upgrades', *Inside CIO*, **10**(7), 24–26.
Sommerville, I. (1992) *Software Engineering*, 4th edition, Addison-Wesley Publishing Co., Wokingham, UK, 649 pp.

Authors' biographies:

Duane W. Hybertson is a Lead Scientist in the Software Engineering Center at The MITRE Corporation in McLean, Virginia, where he investigates software engineering and technology issues for the Department of Defense of the U. S. Federal Government. Current areas of investigation include software architectures, standards and open systems, and reuse technology. Duane received a Ph.D. in educational research methods and statistics from New Mexico State University in 1974, and an M.S. in computer science from The Johns Hopkins University in Maryland in 1985.

Anh D. Ta is a Lead Scientist in the Software Engineering Center at The MITRE Corporation. Since joining MITRE, he has been providing acquisition support to various command centres with the Department of Defense, and participated in research efforts relating to software reuse, safety-critical software, object-orientated technology and COTS integration. Anh received a B.S. in computer and electronics engineering in 1986 and an M.S. in systems engineering in 1990, both from George Mason University in Virginia, where he is currently pursuing a doctorate with emphasis on approaches for developing and evolving domain models.

William M. Thomas is a Lead Scientist in the Software Engineering Center at The MITRE Corporation, where he is involved with the development and application of tools and techniques to support the assessment of large-scale software systems. His research interests include measurement and modelling of software cost and quality, software product assessment, and software reuse. He received a B.S. in mathematics from Bucknell University and an M.S. and Ph.D. in computer science from the University of Maryland at College Park, Maryland.

